

**CS 543: Computer Graphics**  
**Lecture 2 (Part I): 2D Graphic Systems**

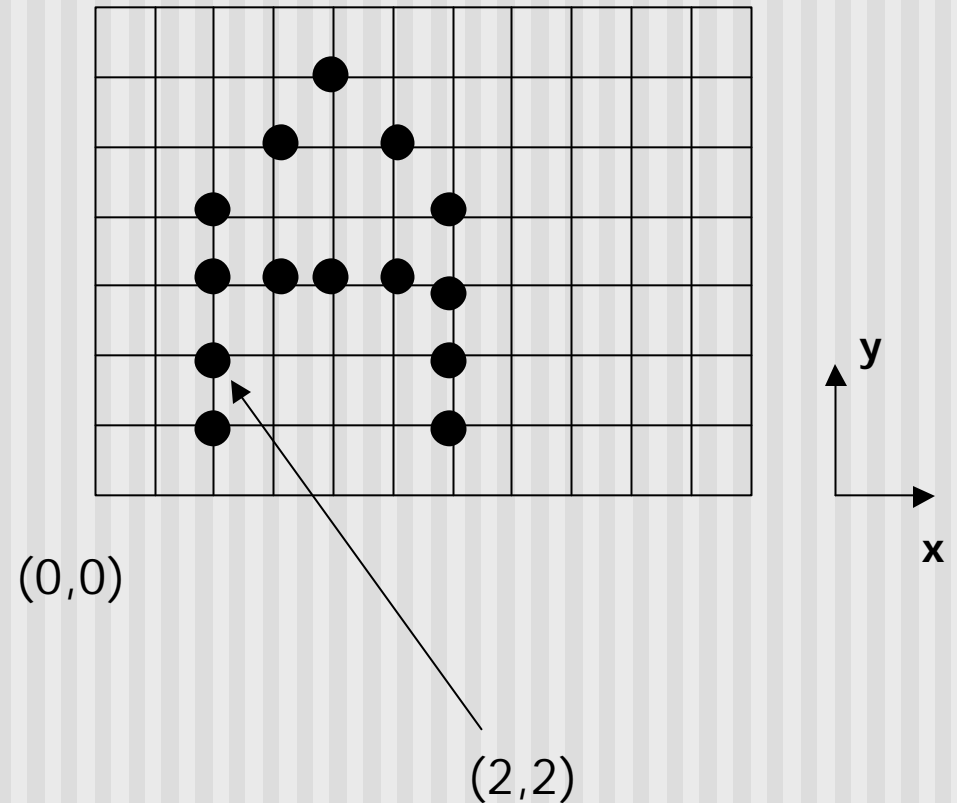
Emmanuel Agu

## 2D Graphics: Coordinate Systems

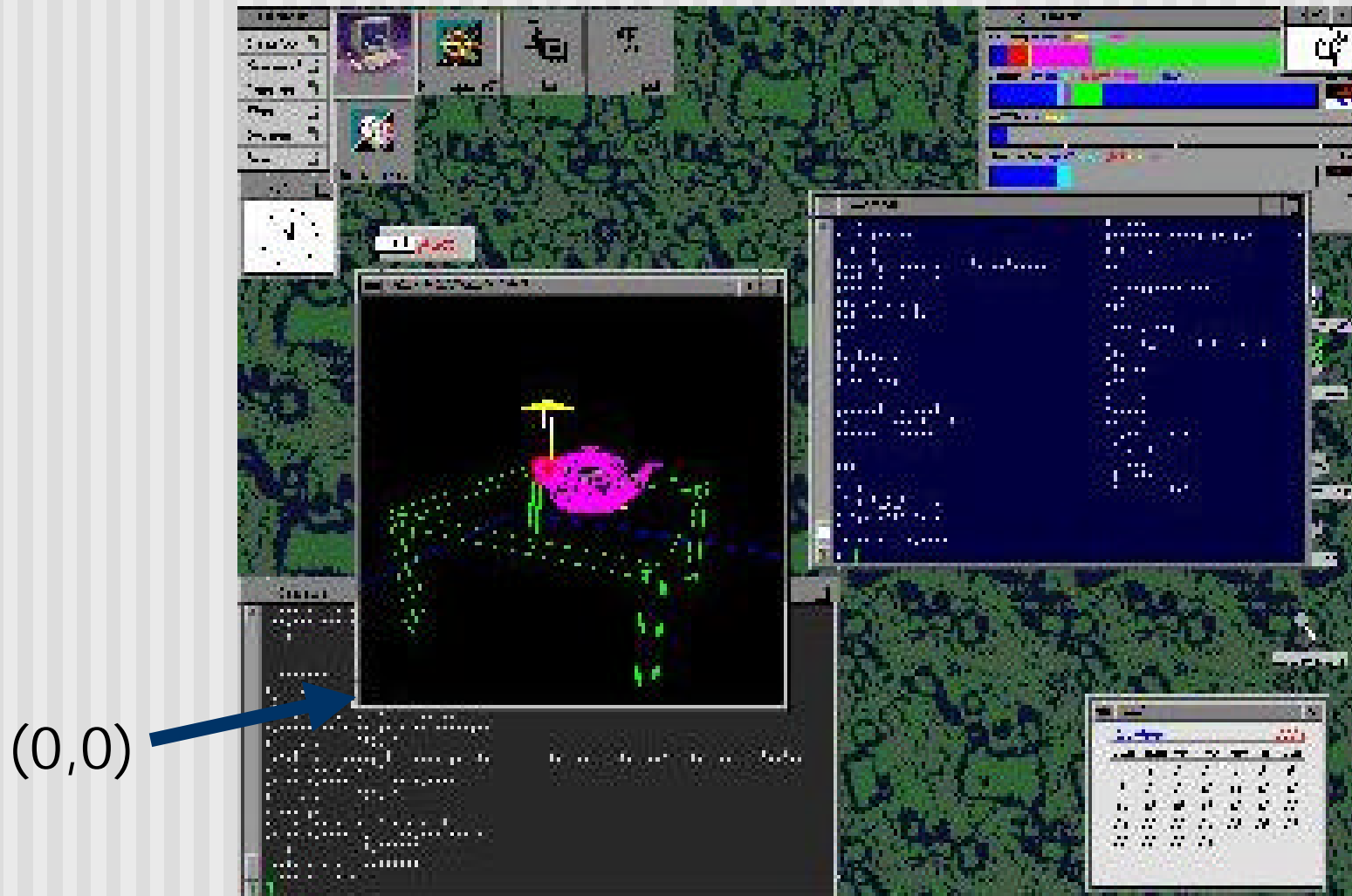
- Screen coordinate system
- World coordinate system
- World window
- Viewport
- Window to Viewport mapping

# Screen Coordinate System

- Screen: 2D coordinate system (WxH)
- 2D Regular Cartesian Grid
- Origin (0,0) at lower left corner (OpenGL convention)
- Horizontal axis – x
- Vertical axis – y
- Pixels: grid intersections



# Screen Coordinate System

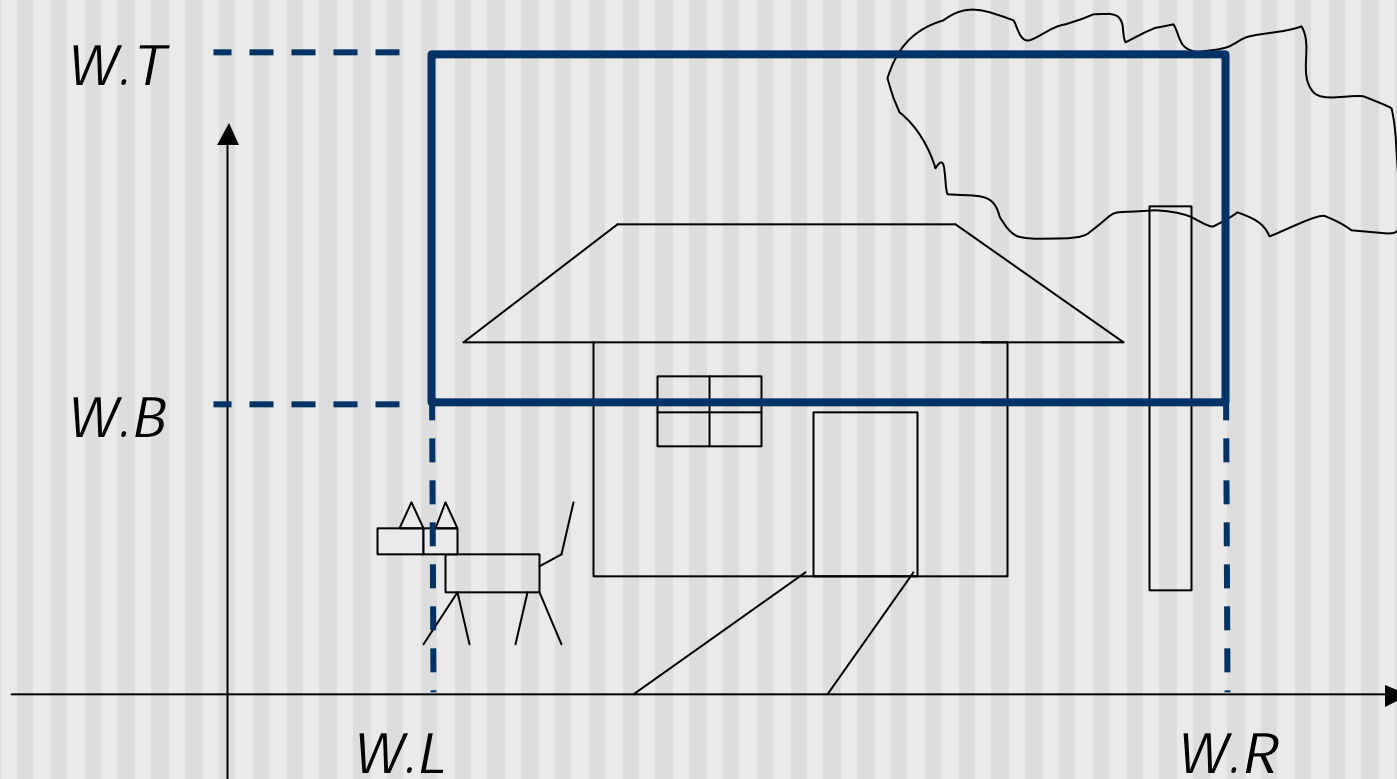


# World Coordinate System

- Problems with drawing in screen coordinates:
  - Inflexible
  - Difficult to use
  - One mapping: not application specific
- World Coordinate system: application-specific
- Example: drawing dimensions may be in meters, km, feet, etc.

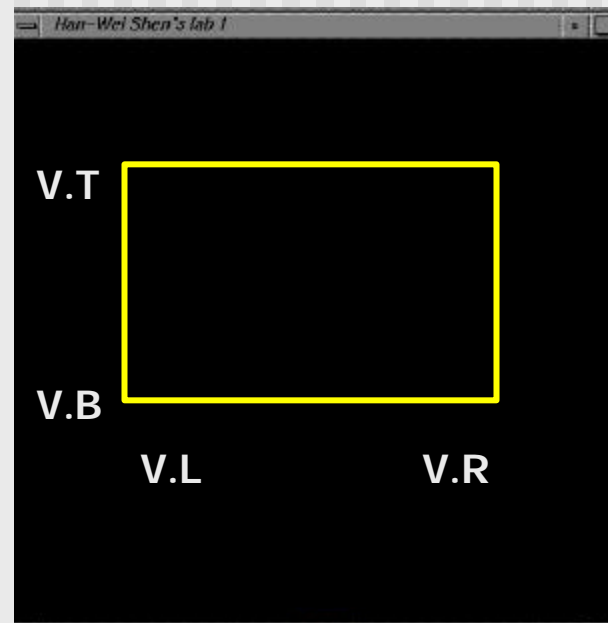
## Definition: World Window

- World Window: rectangular region of drawing (in world coordinates) to be drawn
- Defined by  $W.L$ ,  $W.R$ ,  $W.B$ ,  $W.T$



## Definition: Viewport

- Rectangular region in the screen used to display drawing
- Defined in screen coordinate system



# Window to Viewport Mapping

- Would like to:
  - Specify drawing in world coordinates
  - Display in screen coordinates
- Need some sort of mapping
- Called Window-to-viewport mapping
- Basic W-to-V mapping steps:
  - Define a world window
  - Define a viewport
  - Compute a mapping from window to viewport



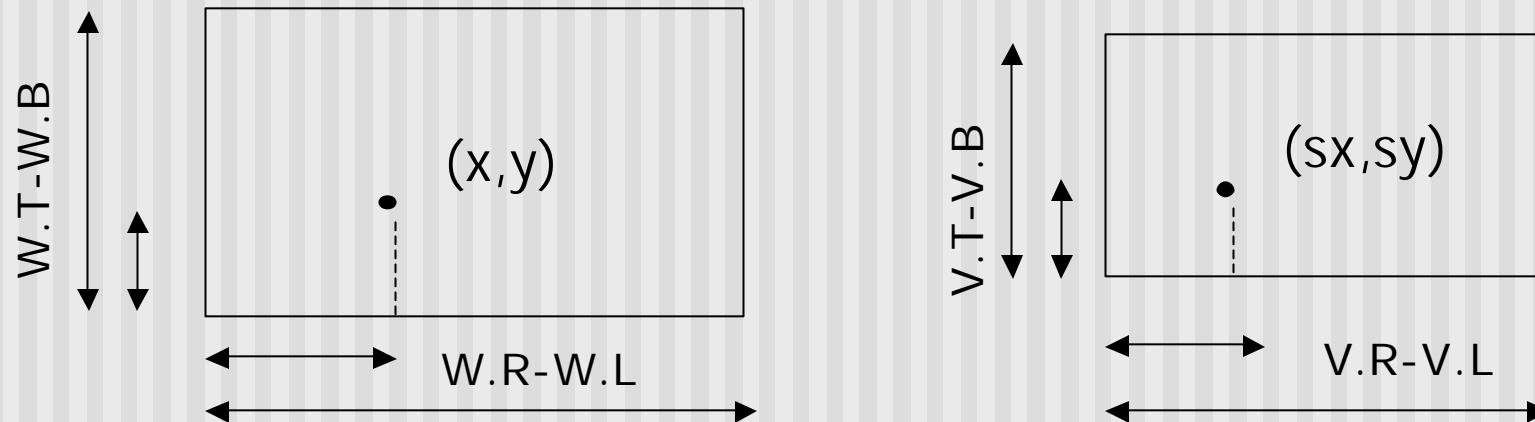
## Window to Viewport Mapping (OpenGL Way)

- Define window (world coordinates):
  - `gluOrtho2D(left, right, bottom, top)`
  - **Side note:** `gluOrtho2D` is member of `glu` library
- Define Viewport (screen coordinates):  
`glViewport(left, bottom, right-left, top-bottom)`
- All subsequent drawings are automatically mapped
- Do mapping before any drawing (`glBegin( )`, `glEnd( )`)
- Two more calls you will encounter to set up matrices:
  - `glMatrixMode(GL_PROJECTION)`
  - `glLoadIdentity( )`
- Type in as above for now, will explain later
- Ref: Hill Practice exercise 3.2.1, pg 86

## Window to Viewport Mapping (Our Way)

- How is window-to-viewport mapping done?
- Trigonometry: derive Window-to-Viewport mapping
- Basic principles:
  - Calculate ratio: proportional mapping ratio (NO distortion)
  - Account for offsets in window and viewport origins
- You are given:
  - World Window: W.R, W.L, W.T, W.B
  - Viewport: V.L, V.R, V.B, V.T
  - A point  $(x,y)$  in the world
- Required: Calculate corresponding point  $(s.x, s.y)$  in screen coordinates

## Window to Viewport Mapping (Our Way)



$$\frac{(x - W.L)}{W.R - W.L} = \frac{sx - V.L}{V.R - V.L}$$

$$\frac{(y - W.B)}{W.T - W.B} = \frac{sy - V.B}{V.T - V.B}$$

## Window to Viewport Mapping (Our Way)

Solve for  $S_x$ ,  $S_y$  in terms of  $x$ ,  $y$ :

$$\frac{(x - W.L)}{W.R - W.L} = \frac{S_x - V.L}{V.R - V.L}$$

$$\frac{(y - W.B)}{W.T - W.B} = \frac{S_y - V.B}{V.T - V.B}$$

$$S_x = \left( \frac{V.R - V.L}{W.R - W.L} \right) x - \left( \frac{V.R - V.L}{W.R - W.L} W.L - V.L \right)$$

$$= Ax - A(W.L) - V.L$$

$$S_y = \left( \frac{V.T - V.B}{W.T - W.B} \right) y - \left( \frac{V.T - V.B}{W.T - W.B} W.B - V.B \right)$$

$$= By - B(W.B) - V.B$$

## Window to Viewport Mapping (Our Way)

Solve, given the formulas:

$$S_x = A_x - \left( A (W.L) - V.L \right)$$

$$S_y = B_y - \left( B (W.B) - V.B \right)$$

What is  $(S_x, S_y)$  for point  $(3.4, 1.2)$  in world coordinates if:

$$W = (W.L, W.R, W.B, W.T) = (0, 4, 0, 2)$$

$$V = (V.L, V.R, V.B, V.T) = (60, 380, 80, 240)$$

## Window to Viewport Mapping (Our Way)

Solution:

$$S_x = Ax - (A(W.L) - V.L)$$

$$A = \frac{V.R - V.L}{W.R - W.L}$$

$$S_y = By - (B(W.B) - V.B)$$

$$B = \frac{V.T - V.B}{W.T - W.B}$$

$$S_x = 80x + 60 = 332$$

$$S_y = 80y + 80 = 176$$

Hence, point (3.4,1.2) in world = point (332,176) on screen

## References

- Hill, 3.1 – 3.3, 3.8